Another Philosophy for Digital Meteor Scatter

by Klaus von der Heide, DJ5HG

1. The Motivation

Scattered pings from underdensely ionized meteor trails are short and usually rare. The footprints on Earth mostly are stripes a few tenths of kilometers wide, and they move at speeds comparable to those of the meteors or even much faster. The chance to loose information from station **A** by simultaneously receiving another station **B** is negligible - even if all European stations are TXing at the same time on the same frequency. The only problem is the tropo-propagation. But that is quite an other point. The question therefore is: Why do most skeds run outside the calling frequency? Why are skeds necessary at all? Why not simply call CQ on the calling frequency and wait for replies?

QSY is not required by the characteristics of the meteor scatter channel and not by the spacial distribution of the radio amateurs. It is caused by the characteristics of the WSJT program [1]. It is this cognition that led the author to search for a quite different digital meteor scatter mode. FSK441 has only one decoder window for everything which looks like an FSK441-signal. It is not easy in a meteor shower to pick out what belongs to your own QSO. Indeed, running the QSO on an own frequency not only reduces the amount of received signals, it also increases the confidence in the decoded information. A received RRR without further information is pure garbage on the calling frequency while it confidently confirms your QSO, if received on your personal QSO-frequency.

The confidence into the received information is an important feature. A real digital communication provides the end-user with correct information. If there is any doubt, the decoded result must be suppressed. In this sense, Packet Radio and Pactor2/3/4 are real digital modes, while PSK31 and FSK441 leave the final decision upon the acceptance to the end-user. So the decision, at least partly, is made outside the automatic communication system. That characterizes the analog modes.

The usual standard procedure of a meteteor scatter QSO is a ***Finite State Machine*** (FSM) with six states. In a QSO two such FSMs are connected via an extremely unreliable radio channel. The initial intention of the author was to design a digital mode for meteor scatter on 144 MHz with reliability of the decoded information as high that both FSMs can run automatically without user interaction. It will be shown in the final chapter that the new mode PSK2k fully reaches this goal.

2. The QSO as a Finite State Machine

The standard procedure for meteor scatter uses six different messages. Some simple rules determine which of these messages has to be sent at a time. The different possibilities where we actually can be in this procedure are called ***states***. Such a system with a finite number of possible states and deterministic rules about the transitions between states is called a ***Finite State Machine*** (FSM). A usual description of an FSM is the ***state diagram***. The state diagram of the standard procedure is shown in figure 1. Transitions for cases where the machine obviously failed because of impatiency of an operator (as in the case when I transmit **B de A R report**, but receive **CQ de B**) are not shown for simplicity.

Running a QSO means to connect two such FSMs via the very unreliable meteor scatter radio channel as shown in figure 2. The communication system consisting of the channel and some hardware and software around it, has the task to more or less completely ***hide the unreliability*** of the channel, i.e. it is the task of this system to make the decision which of the 6 different message types has been received from the other end. This makes clear that the standard procedure is a slave of the communication. It really is a simple machine that depends solely on the decisions of the communication system. As a consequence, the probability to run a QSO sucessfully until the confirmation only depends on the reliability of the communication system. It will be discussed in later chapters how this can be achieved.

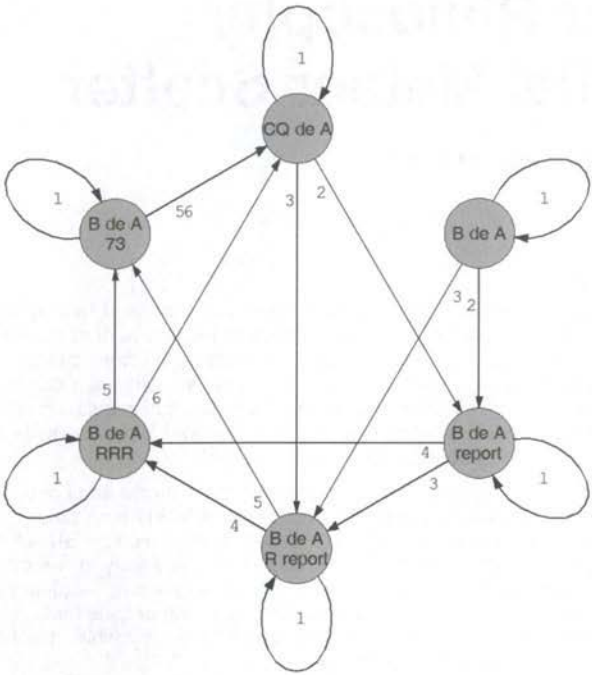


Figure 1. The standard QSO-procedure as a Finite State Machine. The grey circles are the possible states of station A. The actual state of a station is given by the actual transmission. The transitions from one state to another are initiated by (reliable) receptions from the other station B. These transitions are drawn as arrows. The numbers at the arrows denote the condition under which the transition will happen. The possible received messages are:

1 no reception of the following types or reception of the same type which led to this state

2 A de B

3 A de B report

4 A de B R report

5 A de B RRR

6 A de B 73

3. The Problem with the Bitrate

To filter the own QSO-messages out of the decoded output needs (1) information upon both callsigns in the QSO and (2) very high confidence in this information.

FSK441 encodes every character into three sequential symbols of a 4-FSK transmitted with 441 Baud. The character-rate therefore is 441/3 = 147 characters/second. A typical ping of length 0.1 s then can communicate 147*0.1 ≈ 15 characters which is tight for two usual callsigns plus report. And even more: There is no space for any error correcting code plus checksum etc.. If an alphabet of 43 characters is used, then each character corresponds to $\log_2(43)$ = 5.43 bits. Therefore the bitrate of FSK441 is 147*5.43 = 797.7 bits/s. For a digital mode with an error correcting code and a sufficient checksum, the datarate on the radio channel must at least be twice of that.

For his first try to reach such rates the author used GMSK. This retains the advantage of a constant amplitude of the TX-output. The results were decouraging, because simulated weak pings could not be decoded by two reasons: Phase distortions prevented coherent detection, and GMSK has not enough sensitivity. So I decided to give up the demand for constant TX-output. And indeed, doublesideband modulation with 2000 bits/s at a bandwidth of 2000 Hz, but at half the average transmitted power satisfied all other demands:

(1) low error rate of the raw data

(2) high datarate of 2000 bits/s

(3) low bandwidth of 2000 Hz fitting the needs of a usual SSB-transceiver

That was the real starting shot for the design of a new digital mode for meteor scatter in December 2010.

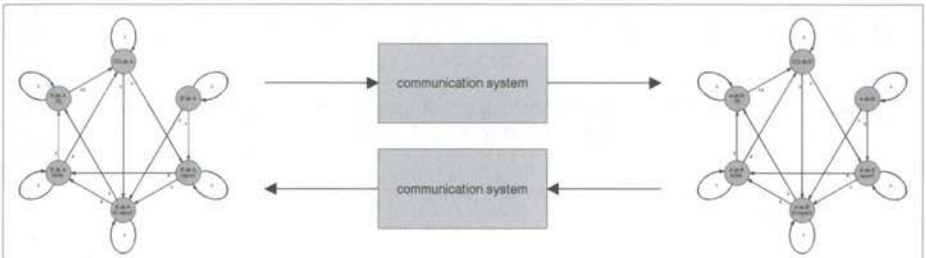


Figure 2. The complete QSO-machine consists of the FSMs at both ends and the communication systems for both directions. The two communication systems are mutually switched ON/OFF for the period length. The time is synchronized to UTC at both ends. The task of the communication systems is to transport the actual state of a station reliably to the other. It is not the 'RRR' as some letters a station wants to get as the final confirmation of a QSO, it is the reliable information that the other station really is in the state of sending the 'RRR'. This received state information is the input into the own FSM. And the ***decision*** upon the possible transition to a new state only is ***based on this received information*** about the state at the other end. If any state information is sent via chat in the internet, then the QSO is made via the internet. The communication system then is unnecessary, because it cannot increase the knowledge upon the state at the other end. This knowledge is the basis of a dialog. The knowledge whether there is a ping displaying some RRRs or not, when I already know that the other end sends RRRs is useless.

4. The Modulation

The digital-to-analog conversion of the bitsequence is done by replacement of every bit by a fixed pulse-function multiplied with +1 if the bit is 1 and with -1 if the bit is 0. The pulse-function is sampled at a standard sampling rate.The pulse function used in PSK2k is a sinc function as shown in Figure 3. Figure 4 explains how the PSK2k-signal is generated from a given bitsequence by adding all these individual pulse functions. The resulting signal (figure 4b) now has to be modulated.

The signal of figure 4b is multiplied with a sine-wave of 1500 Hz. The result is the Double-Side-Band signal shown in Figure 5. This is the audiosignal that is shifted by the SSB-transceiver to the target frequency. It is called a BPSK signal because the phase is 0° where the binary values (the red dots in figure 4) are +1, and 180° where they are -1. But in reality, it is an amplitude modulation with suppressed carrier.

The spectrum of the baseband signal of figure 4b reaches from 0 to 1000 Hz with a sharp falling edge at 1000 Hz. Therefore the amplitude modulated signal of PSK2k is rectangular, and it extends from 500 Hz to 2500 Hz with extremely sharp edges. This bandwidth is less than that of FSK441 - on the premises that the PA is not run into power limitation.

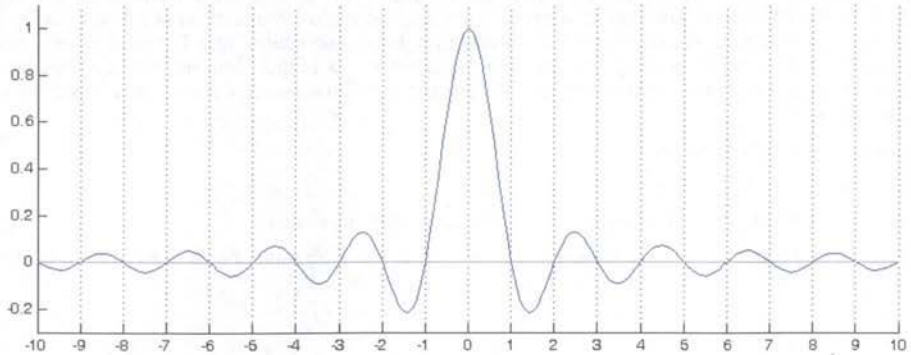
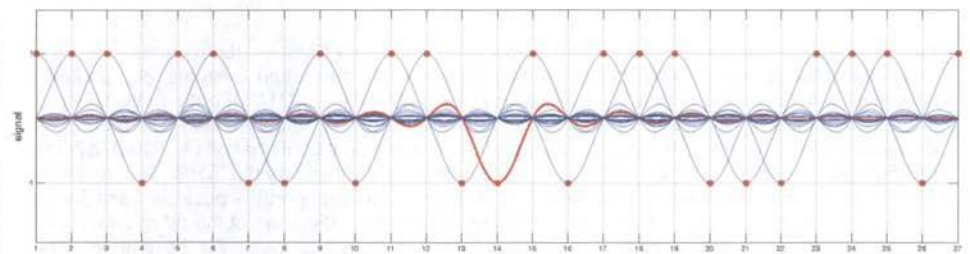
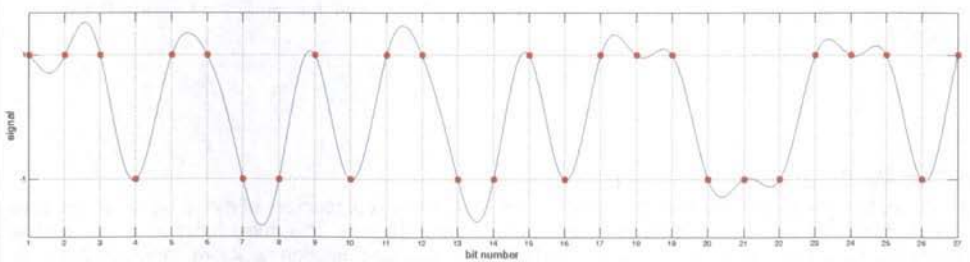


Figure 3. The sinc-pulse. It is infinitely long to both sides with slowly decreasing amplitude. It has zeros at k = ±1, ±2, ±3, ...



a.



b.

Figure 4. Generation of the PSK2k baseband signal. The example bits [1 1 1 0 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 1 0 1] are replaced by the values v=±1. These values are replaced by v times the sinc-function. All these functions are shown as blue lines in figure 2a except for one which is colored in red for demonstration. It is obvious that all other functions have a zero where one function has ist value 1 or -1. The sum of all sinc-functions in figure 2b therefore goes through all the values v which are marked as red dots.

The horizontal spacing of the bits (1.0 in the figure) corresponds to 0.5 ms because of the bitrate of 2000 bits/s. The PSK2k-signal is an analog signal. But the PSK2k program generates it at a user-chosen sampling rate. If the sampling rate is 32000 for example, the spacing is 16 samples.

5. The Very First Real-Life Test

The basic concept was chosen in a few weeks. To get confidence that the enormous programming effort for a reliable PC-based mode would not be in vain, I generated an audio track for a real-life test. Peter, SM2CEW, was so kind to transmit this test on evening Jan 26, 2011. The test was negative. I simply got not a single decode, I also didn't hear any ping. We tried again the next day. Now I got a few weak pings, all decoded with a frequency offset very near to 200 Hz. In fact, I had set the limit for the frequency offset to ±200 Hz. I increased this limit to ±300 Hz and decoded the record of the previous day again. It gave two decodes of tiny pings with frequency offset slightly above 200 Hz - not much, but very encouraging, because really weak pings were completely decoded and no garbage produced within the test time.

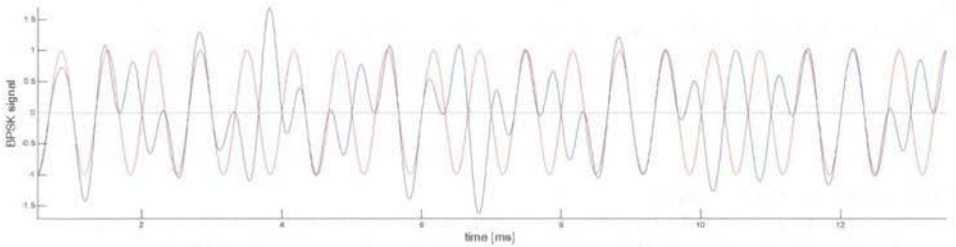


Figure 5. The BPSK-signal (blue) for the same binary data as in figure 4, and the 1500 Hz carrier (red).

6. The Basic Idea

All the basic ideas were born while the author wrote three articles for a special magazine [2,3,4] - two of them concerning FSK441. FSK441 uses the characters of the alphabet as the basic information packets. The advantage of these very short packets of only 6.8 ms length is that a ping can be used from it's rising out of the noise until it vanishes there. But there are several disadvantages:

(1) There is no chance for error detection and error correction.

(2) The output is not aligned (a **CQ DJ5HG** may look like **HG CQ DJ5**) .

(3) The received information cannot automatically be associated with actually running QSOs.

On the other hand, with a bitrate of 2000 bits/s we could transmit 200 databits in a typical pinglength of 0.1s. Since two callsigns plus some QSO-information is much less than 100 bits, there is much freedom for error correction and error detection, if a packet of about 200 bits is used.

But it is not sufficient, simply to receive the databits. We also must know, which of the databits is the first bit of the packet. The simplest way is to use a fixed bitsequence as a header. As a consequence, the effective bitrate then is reduced by the value (length of header) / (length of packet).

A CQ-call only transmits one callsign. So a CQ contains less information than a reply. If a fixed packet format is used, then there are some information bits free for other information. Extensive discussions with Roger, GW4WND, led to many details of the current implementation of the new digital mode PSK2k presented in the following chapters. One of these details is the transmission of the QTF with a CQ. PSK2k can use three period length: 15s, 30s, and 60s. The period also is sent with a CQ.

When a QSO is established - after exchanging the callsigns - the messages for report, roger, and 73 contain a few bits only. The larger portion then is the address information. This address information need not to be transmitted in the same encoding as it is done to exchange the callsigns. A basic idea of the design of PSK2k is to use a binary representation of the addressed callsign as the packet header in all messages of a QSO following the exchange of the callsigns. As a consequence, PSK2k uses several different packet formats. The principle of packet formation is shown in figure 6.

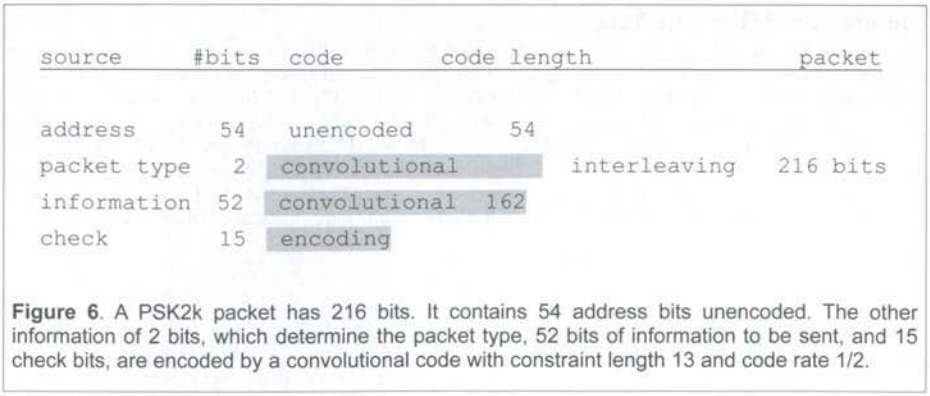

source #bits code code length packet
address 54 unencoded 54
packet type 2 convolutional interleaving 216 bits
information 52 convolutional 162
check 15 encoding

Figure 6. A PSK2k packet has 216 bits. It contains 54 address bits unencoded. The other information of 2 bits, which determine the packet type, 52 bits of information to be sent, and 15 check bits, are encoded by a convolutional code with constraint length 13 and code rate 1/2.

7. The Address Bit Patterns

All PSK2k-packets contain 216 bits. 54 bits are address bits used for synchronization, and 162 bits are a codeword. The address bits are interleaved with the codeword: Bits 1,5,9,13,... of the 216 packet bits are the address bits. The spreading of the address bits all over the packet guarantees that even a very short ping will contain enough address bits for a successful synchronization.

PSK2k uses two different binary address types of 54 bits:

The ***general address*** is used for all transmissions addressed to everybody, i.e. CQ calls, QRZ calls, and QSTs. The general address has the unique bitsequence

`001111011110010101000100111011000101011000010001001011`

The private ***address*** is used to address a target callsign. It is unique for every callsign. The address for DJ5HG for example is

`000011110101100101010110101001110010001110101010110011`

The encoding of the address is specified in the PSK2k-documentation [5].

8. The Error Detection

Error correction and ***error detection*** are different aspects, and they need different solutions. An error correcting code is used to maximize the rate of information bits over a channel, which in our case of meteor scatter pings means to maximise the information transported by a ping. If the ping is too short or too noisy, then the error correction fails, and the decoded result is worse than it would be without using such a code. It is the purpose of an error detecting code, to separate correct and false decodes confidently. The lack of an additional error detecting code can be very irritating as in JT65. The deep search decoding method even contradicts any such code. Or in other words: Deep search cannot be used if a communication system does what it should do: to give reliable results.

Error Detecting Codes usually are systematic codes. A systematic code is one that starts each codeword with the information bits unmodified. The additional bits are called parity bits or check bits. These bits are generated from the information bits by a fixed algorithm. The receiver first decodes the outer error correcting code. Then it simply generates the parity bits from the decoded information bits and compares them to the decoded parity bits. If there is any difference, the message is discarded.

The capability of error detection is limited. If only one parity bit is used then on average every second faulty message would be accepted by random. In the case of n parity bits the probability of accepting a faulty message is not lower than 2^{-n}.

The error detection in PSK2k generally is based on residual codes. This means that the information bit array is interpreted as a natural number z. The parity bits are computed as the binary representation of the residual of the division of z by a fixed quotient r.

Examples with r=31.

binary information arrays	0000010001	1000111101	1111010010
z	17	573	978
remainder	17	15	17
parity bits	10001	01111	10001

If r is a prime number then the chance to get acceptable parity bits in case of a received random bit sequence is 1/r. PSK2k uses three different values of r: 32051, 8009, 4093.

9. The Error Correction

PSK2k uses two different convolutional codes for error correction. These codes and details of the encoding are described in [5]. Both codes result in 162 bits. The 54 address bits are inserted in a regular manner leading to the packet of 216 bits as described in chapter 7.

10. Data Interleaving

The convolutional code for CQ-calls and replies generates two sets of 81 bits (162 in total). The bits of these sets are counted from 1 to 81 and colored green and cyan in figure 7. Data interleaving means that the sequence of the databits sent on the radio channel is other than their original sequence. Figure 7 shows the arrangement of the bits in the packet.

Any set of 11 sequential bits of a PSK2k-packet contains information upon every bit of the encoded information. This does not mean that the transmitted information can be reconstructed from 11 sequencial bits, which corresponds to 5.5 ms. But it means that all such short sequences influence every information bit by the same strength. If for example, a ping has a very sharp peak, as it occurs on 70 cm, then the few databits within the peak give benefit to each bit of the finally decoded information. So it is completely arbitrary how the received energy is distributed within a ping. Only it's sum compared to the noise power determines upon decodability of the received packet. This is demonstrated in figure 8.

1	1	19	37	2	55	73	10	3	28	46	64	4	2	20	38	5	56	74	11
6	29	47	65	7	3	21	39	8	57	75	12	9	30	48	66	10	4	22	40
11	58	76	13	12	31	49	67	13	5	23	41	14	59	77	14	15	32	50	68
16	6	24	42	17	60	78	15	18	33	51	69	19	7	25	43	20	61	79	16
21	34	52	70	22	8	26	44	23	62	80	17	24	35	53	71	25	9	27	45
26	63	81	18	27	36	54	72	28	10	28	46	29	64	1	19	30	37	55	73
31	11	29	47	32	65	2	20	33	38	56	74	34	12	30	48	35	66	3	21
36	39	57	75	37	13	31	49	38	67	4	22	39	40	58	76	40	14	32	50
41	68	5	23	42	41	59	77	43	15	33	51	44	69	6	24	45	42	60	78
46	16	34	52	47	70	7	25	48	43	61	79	49	17	35	53	50	71	8	26
51	44	62	80	52	18	36	54	53	72	9	27	54	45	63	81				

Figure 7. Arrangement of the address bits (red), and the bits of the two sets of 81 data bits of the convolutional encoder (green and cyan) within the PSK2k-packet. The bitsequence of the packet must be read row-wise in this diagram.

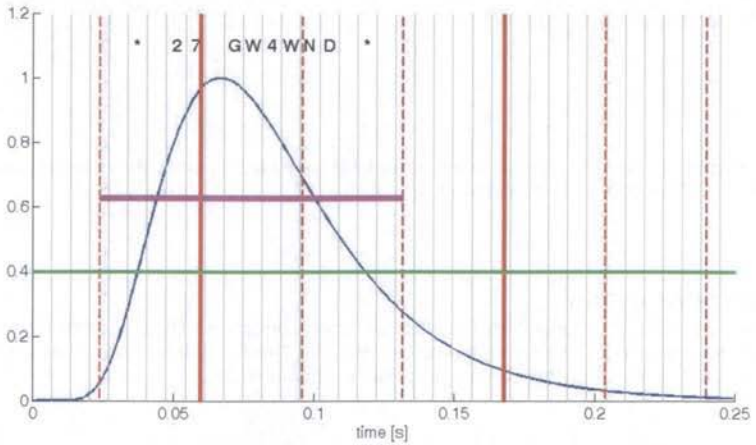


Figure 8. The amplitude of a typical ping on 144 MHz as a function of time (blue line). The noise level is indicated by the green line. The vertical gray lines define the grid of sequential characters of FSK441. Only 11 characters are above the noise level. So a message like GW4WND DJ5HG 27 27 will only be received partly. A PSK2k-packet is 0.108 seconds long as indicated by the two bold red vertical lines. But the PSK2k-receiver also finds packets shifted by ±1/3 of a packet as indicated by the dashed red vertical lines. The best synchronization is selected to be decoded. The bar in magenta color shows the best choice and the level of the mean amplitude. This mean level is well over the noise. Therefore the packet can be decoded to a very reliable complete message.

11. The Source Codes

PSK2k uses several different source code formats, which all are discussed in the documentation [5]. To give some insight, we only discuss here the addressed contest message with report, roger, contest number, locator, QTF, power, and antgain. The final bitpattern of the information has 54 bits.

This bit array starts with the message type bits 10. The information itself is encoded into the following 52 bits:

Let be

R the roger bit (0: no roger; 1: roger)

s a report index (starting index is 0).

The list of possible reports is { 0dB, 3dB, 6dB}

i the index into the list of the number of decodes (starting index is 0).

The list of possible number of decodes is { 1, <4, <8, <16, >15 }

n the contest QSO-number (0<n<1416)

q the index into the list of possible QFT-values in degrees (starting index is 0).

The list of possible QTF-values is the QTF in steps of 5 degrees { 0°, 5°, 10°, ... , 355°,

'round horizontally polarized', 'round vertically polarized')

L an array of 6 indices that defines the Maidenhead locator (starting indices are 0).

The indexed alphabet of the first two indices is ABCDEFGHIJKLMNOPQR

The indexed alphabet of the last two indices is ABCDEFGHIJKLMNOPQRSTUVWX

The indexed alphabet of the two indices between is `0123456789`

p an index which specifies the actual transmitter power (starting index is `0`).

The indexed list is `{ 10W, 25W, 50W, 100W, 250W, 500W, 1kW, 2kW }`

g an index which specifies the actual antenna gain (starting index is `0`)

The indexed list is `{ 0dB, 3dB, 6dB, 9dB, 12dB, 15dB, 18dB, 21dB }`

The information bit array now is assembled:

`bits(1...2)` `1 0`

`bit(3)` R

`bits(4...22)` binary representation of `370*(n-1) + 5*q + i`

`bits(23...48)` binary representation of the following number:

`18662400*s + 777600*L(6) + 32400*L(5) + 3240*L(4) + 324*L(3) + 18*L(2) + L(1)`

`bits(49...51)` binary representation of p

`bits(52...54)` binary representation of g

Example: Le the actual data be no roger, report is 3dB, only 1 ping decoded, contest number is 35, QTF is 65°, the locator is JO53IM, transmitting power is 600W, antenna gain is 17dB. The above variables then are:

`R=0, s=1, i=1, n=35, q=65/5=13, L=[ 9, 14, 5, 3, 8, 12 ], p=5, g=6.`

With these values we finally get the bit array

`100000001100010110011001101011110100011111010001101110`

15 parity bits are generated by the following procedure:

(1) The binary patterns of both callsigns in the QSO are appended to the binary pattern above.

(2) The resulting binary number is divided by `r=32051`.

(3) The remainder of the division (15 bits) is appended to the above pattern giving 69 bits.

These 69 bits are encoded with the convolutional code resulting in 162 bits. The interleaving as shown in figure 7 finally leads to the 216 packet bits.

The baseband signal is generated from this packet following figure 4. Multiplication with the sinewave of the carrier of 1500 Hz finally leads to the transmitted signal.

12. The Graphical User Interface

The Graphical User Interface (GUI) of PSK2k as shown in figure 9 is similar to that of WSJT with FSK441: There are displays of the signal and the spectrum, there is the standard procedure to select a message manually, and there are some buttons for program control. But there are large differences in the details:

(1) There are three decoder windows instead of one:

The upper window displays all messages addressed to all users, i.e. CQ calls, QRZs, QSTs.

The lower window displays all messages of the actual QSO.

The window between displays calls to this station which are not replied until now.

(2) There are two buttons *normal* and *contest* to generate the standard procedures for the normal mode and for the contest mode.

(3) There is a bar of four buttons *manual* *QSO* *reply* *CQ* to select the different automatic modes of PSK2k.

The PSK2k UserGuide [6] offers a detailed description of all features.

13. The Automatic Operation

If one already for half an hour had sent RRR to confirm a QSO, but got nothing back, then impatiency reaches a level, where one decides to stop the transmission. After this decision you may get via chat in the internet 'I still need your RRR'. What to do now? If you decide to start with RRR again, then this decision solely is based on information transfer via the internet. Indeed, the QSO via the radio path is completely lost, when you make the decision to stop the transmission. He simply failed to communicate his QSO-state (A de B R report) within a time interval given by your personal impatiency. If you now want to send him the RRR, then send it via the chat.

The FSMs as shown in figure 1 and figure 2 base their decisions of state transitions solely on received messages. The scenario mentioned above really happens. Or the other station gets the RRR immediately after you started sending them. He is happy and switches the station off. My FSM then would send the RRR until the end of the universe. This makes clear that the simple FSM must be extended by something that corresponds to my impatiency. PSK2k uses two timers for this purpose. All timers can individually be set in minutes:

(1) A timer limits the time to transmit the same packet. The timer is resetted by every decode belonging to the running QSO. If the TX is stopped by this timer, it immediately starts again, if any message belonging to the running QSO is decoded.

(2) A timer limits the time to keep up the running QSO. The timer is resetted by every decode belonging to the running QSO. If the QSO is stopped by this timer, it can only be resumed manually.

14. The PSK2k-Receiver

The receiver is the most complex subsystem in the PSK2k-program. We will discuss only some aspects here. The first stage of the receiver is a Hilbert filter, which transforms the real input signal to a complex signal. The next stages are the very simple birdie-blanker and the noise blanker. Then the demodulation follows. It is briefly explained in the following chapter.

14.1. Detecting the Carrier Frequency and Demodulation

Let $m(t)$ be the baseband signal shown in figure 4b and let the carrier frequency and the phase be f and φ. f and φ are unknown at the receiving end. The received signal - for simplicity without noise here - has the form (amplitude-modulated carrier of frequency f and phase φ).

$$x = m(t) * \exp(i (2 \pi f t + \varphi))$$

The receiver computes the square of all samples:

$$x^2 = m^2(t) * \exp(2 i (2 \pi f t + \varphi)) = m^2(t) * \exp(i (2 \pi (2f) t + 2\varphi))$$

$m(t)$ is a real function. It oscillates in the region -1.3 ... +1.3. The modulated signal x has a suppressed carrier of frequency f. But the square of $m(t)$ is positive. Therefore, the square of x is an amplitude-modulated signal with carrier of frequency $2f$. The detection of the carrier frequency of an amplitude modulated signal is very robust. The PSK2k-receiver computes the Fast Fourier Transform (FFT) of the square of x. The maximum peak within this spectrum is at $2f$. Fortunately the FFT also gives 2φ. With these values the receiver then computes

$$x * \exp(-i (2 \pi f t + \varphi)) = m(t) * \exp(i (2 \pi f t + \varphi)) * \exp(-i (2 \pi f t + \varphi)) = m(t)$$

which is the demodulated real baseband signal.

We have neglected a problem here: if we know 2φ, which is in the range of $0 ... 2\pi$ we can reconstruct φ only to $\varphi = (2\varphi)/2$ in the region $0 ... \pi$. But φ in reality also could be in the region $\pi ... 2\pi$ which would lead to 2φ in the region $2\pi ... 4\pi$ what is the same as $0 ... 2\pi$. The consequence of this ambiguity is that we do not know the sign of the reconstructed $m(t)$, i.e. we finally get a sequence of bits, but we do not know whether we should take them as they are or invert all bits.

But the sign of all bits of the address is known. If the selected maximum correlation value is negative, then the sign is -1, otherwise +1.

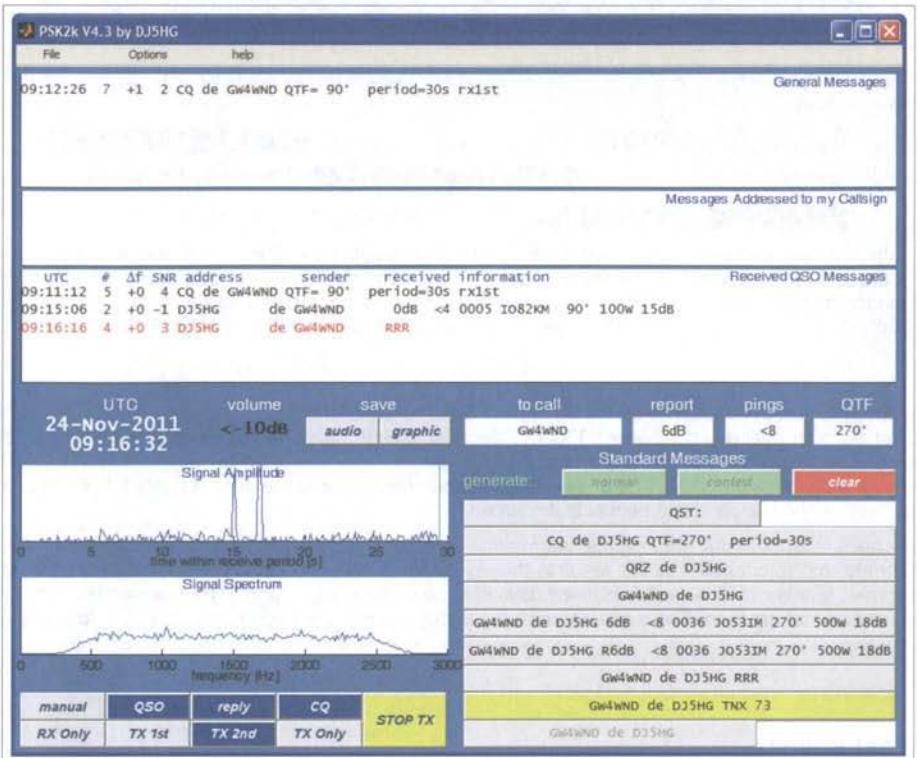


Figure 9. The Graphical User Interface (GUI) of PSK2k. Decoded messages are displayed in three decoder windows: The upper window for CQ, QRZ and QST, the window for pending calls to this station waiting for a QSO, and the lower window for received QSO-information. The lower half of the GUI shows the actual standard procedure to the right, and signal displays and control buttons to the left. The GUI can be resized. The fonts then are resized too. The messages shown here are replayed from recordings of tests with Roger, GW4WND.

A single mouse-click on a line in a decoder window copies the line into the clipboard.

In the actual case, a double-click on the decoded CQ-line generated the standard procedure texts and started the QSO. Since the AutoMode CQ is on, the QSO ran automatically. After reception of either 73 or CQ from GW4WND, the program will start a CQ from this station.

Decoded lines of the same information content are overwritten in red color, and the number of decodes # is updated. While the TX is running, the selected TX-message is colored yellow (or red in a simulation).

14.2. Bit-Synchronization and Packet Synchronization

Let the input sampling rate be 16000 samples/second. Then the length of one bit (0.5ms) is 8 samples. The baseband signal m(t) of one packet then has 216*8 = 1728 samples. In respect to a simple realization of the packet shift of ±1/3 packet length as shown in figure 8, the receiver analyses the concatenation of two packets at a time. So the input to the receiver is an array of 2*1728 = 3456 samples. This array is reshaped to a matrix of size (3456/8/4) * (8*4). This matrix is filtered column-wise with all 6 address patterns yielding 12*32=384 filtered signals of length 108. The absolut maximum value of each signal points to the last packet bit, and the search for the maximum of all these absolut maximum values differentiates between general address and private address and also between the thee possible packet shits, and it determines which samples should be taken for the read-out of the bits.

The number 6 of different address patterns is the product of 2 for the two different addresses, the genneral address and the private address, and 3 for the three possible shifts. For example, the three general addresses rotated in steps of 18 bits are:

shift	pattern
0	001111011110010100010011101100010101100001000100101
-18	000100111011000101011000010001001011001111011110010101
+18	011000010001001011001111011110010101000100111011000101

The output of the synchronization stage is an array of 216 real values. They are called soft bits, because they represent the received 216 packet bits, but they are corrupted by noise, and there is no decision to their binary values. If the best correlation was with an address pattern shifted by s bits, then the whole packet of 216 bits is rotated by -4s bits to get the correct bit sequence in the codeword.

14.3. Error Correction, Error Detection, and Source Decoder

A Viterbi decoder decodes the packet. It is this decoder, which makes the decision upon what has been received. The result is a pattern of 69 bits. The first 54 bits now are used to generate 15 checkbits as discussed in chapters 8 and 11. These generated bits are compared to the decoded bits 55...69. If there is any difference, the the complete packet is discarded.

If the check is successful, the source information is reconstructed from bits 1...54 using the reverse algorithm to that of chapter 11. The result is the textstring of the decoded information. Some information on pingtime, number of decodes, frequency deviation (taken from 14.1), and SNR is appended at the front of the decoded information. The resulting textstring is displayed in the selected decoder window as shown in figure 9.

15. Test Results

The first-ever bidirectional QSO in PSK2k-mode was made on Aug11, 2011 between Peter, SM2CEW and DJ5HG. Extensive tests were run with Roger, GW4WND, on Sep12, Oct21, Oct23, Nov4-7, Nov17, Nov20. Most of these tests took many hours, the test Nov4-7 ran continuously over more than 60 hours. Most of the tests were made with at least one of the stations without manual interaction, often simply starting a call to the other (automatic) station manually and then switching to automode leading to a fully automatic system as shown in figure 2.

As an experienced operator, Roger contributed a lot of suggestions to the program control and to the user interface - not to count the number of bugs in the program behavior he found (and I then had to fix).
Some results should be discussed more in detail:

15.1. The Reliability of PSK2k

We had a few false decodes in the first tests. To suppress them further, I increased the significance level of the synchronization considerably. The loss in number of decodes is negligible, but the gain in suppression of false decodes is large. The number of false decodes in continuous operation of more than 200 hours now is one. The rate of false decodes will increase if many stations run PSK2k on the same frequency. But it will be sufficiently low to run PSK2k in automatic mode.

15.2 Automatic Operation

As mentioned above, PSK2k was mostly tested in automatic mode. This allows tests which in manual mode usually are limited by the operator's impatiency. A QSO on afternoon Nov. 5 took 6 hours to be completed. It ran while both operators were busy with other work. In most of the tests we used reduced power, Roger even tested with 10 Watts (which gave decoded results with SNR up to 4dB).

The critical point in automatic operation is the setting of the timers. The timer values must be large enough that transmission (or the QSO itself) are not stopped before the actual message has been received at the other end. On the other hand the timer values should be small, because otherwise the

transmission of a final RRR or 73 is blown for nothing into nirwana, and - that's the important point - the station is not ready for the next QSO, while it tries to finish the last.

A really difficult situation is running 500 Watts at one end A and 10 Watts at the other end B. The well-powered station A may get nothing for hours, while B gets decoded results at intervals of some minutes. A tends to set his timers to large values, say 150 minutes for the TX timer. B tends to reduce the timer values. We really encountered the situation, when A were sending the final RRR and B got it immediately and he started to send 73. But the 73 were never received until they were stopped at B, while A were sending the RRR for hours, because only a 73 or a CQ received from B or the timer at A could have stopped the transmission. Nevertheless, this is not a weakness or even a fault, it is a general problem caused by the two very different communication systems.

In summary, the automatic operation of a station seems to be more reliable than the manual operation. This mainly is caused by the great patiency of an automaton - if the timers are set properly. Indeed, except from a few earlier QSOs that ran into program failures, all QSOs that were successful with the exchange of the callsigns, also were completed.

15.3. Doppler Shift

The PSK2k-receiver needs to reconstruct the carrier (see chapter 14.1). The accuracy of the carrier frequency is limited by physical reasons based on the ping length to about 2 Hz. This is sufficient to measure the Doppler shift caused by the movement of the meteor trail in the ionosphere, if GPS-synchronized TX and RX are used at both ends. We indeed observed small systematic deviation of some Hz at all decodes in a period of an hour and later had different systematic deviations. Since our main goal was to test the mode, we did not analyse these effects further. But it may be an interesting task for future work.

16. Conclusion

The main aspects shown in this paper are:
The overhead of an error correcting code plus interleaving can lead to better decodes of very short pings although the packet length is very much larger than in a character-oriented mode like FSK441 (see figure 8). The reliability of running meteor scatter QSOs until confirmation - especially under difficult conditions like very low transmitter power or extremely rare meteors - seems to be much larger, if the standard procedure is run automatically instead manually by an impatient operator.
This predestinates PSK2k for extreme situations like multi-op operation or expeditions.

Acknowledgements

PSK2k would not exist - at least not in a usable version [7] - if I would not have been encouraged by the tests with SM2CEW and GW4WND. Many thanks for it to Peter and Roger. Especially the many comments and suggestions of Roger are very appreciated.

References

[1] http://www.physics.princeton.edu/pulsar/K1JT/wsjt.html
[2] Klaus von der Heide, 'Digitale Kommunikation: Grundlagen und Grenzen',
CQ DL-Spezial Amateurfunk digital, 2011, pp.4-10
[3] Klaus von der Heide, 'Bauen in Software',
CQ DL-Spezial Amateurfunk digital, 2011, pp.11-15
[4] Klaus von der Heide, 'Meteorscatter mit FSK441',
CQ DL-Spezial Amateurfunk digital, 2011, pp.58-62
[5] http://www.DJ5HG.de/PSK2k_Documentation.pdf
[6] http://www.DJ5HG.de/PSK2k_UserGuide.pdf
[7] http://www.DJ5HG.de/PSK2k.zip